Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 89 (2016) 34 – 42

Procedia Computer Science

Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)

Terminal Authentication in M2M Communications in the Context of Internet of Things

Aditya Kaushal Ranjan* and Muzzammil Hussain

Central University of Rajasthan, Kishangarh, Ajmer 305 817, India

Abstract

Security and Privacy is a major concern in IoT and within terminal authentication is a major issue in M2M communications. Here we have proposed a set of protocols and its associated mechanism for terminal authentication in M2M systems in the context of IoT. It is based on digital signature and randomly generated credentials. The main feature of the proposed protocol is it does not depend on pre-installed keys and other tokens. It is secure and no known attack possible on this as we have shown in informal threat analysis and above all it does not depends on Certification Authority/Trusted third party and consumes very less resources. It is also very efficient and deployable that we have shown through experiments. It meets all requirement of IoT scenario like Heterogeneity, Scalability etc.

Peer-review under responsibility of organizing committee of the Organizing Committee of IMCIP-2016

Keywords: Heterogeneity; IoT; M2M Communications; Scalability; Terminal Authentication; Threat Analysis.

1. Introduction

M2M (Machine to Machine) and IoT (Internet of Things) are the two technologies that are in the core of convenience for modern human lifestyles. Anything in the physical world that is of people and organisational interest to observe and control will be connected and will offer services *via* the internet. M2M and IoT are the results of technological advancement due to decreasing a cost of semiconductor chips and broad deployment of the internet. Emerging reasons are basically to understand physical surrounding of various domains and also in the advancement in another field like sensors, electronic tags, actuators etc. Thus, the machine can be used in a more intelligent manner, often termed as smart objects[1].

M2M refers to that communication between the same type of devices and specific application, all *via* wired and non-wired networks. The application areas like remote monitoring and control of assets or to provide connectivity to remote devices. Traditionally, M2M is more point oriented in approach where devices and applications are highly dedicated to solving a single task. But with the requirement of gathering the information and services from various sources with more flexibility, devices can no longer be application-specific in the same manner as for M2M[1]. So there is a transition from M2M to IoT.

*Corresponding author. Tel.: +91 -9771195196
E-mail address: aditya.k.ranjan@ieee.org

Peer-review under responsibility of organizing committee of the Organizing Committee of IMCIP-2016
doi:10.1016/j.procs.2016.06.006

There are mainly three layers in the IoT *viz.* physical perception, network and application. All have their own security issues and have various challenges like protocol and network security, identity management, privacy, trust, fault tolerance etc.[2]. Authentication is also an important aspect of identity management, so we need to provide some mechanism for achieving authentication. Especially for terminal authentication, perfect security mechanism is needed to detect compromised terminals and to sure that data flow generates by certain entity actually contain what it is supposed to contain[3]. This motivates us to work on this problem and find effective solution.

The paper is organised in following way. Section 2 describes the proposed work. In this section, we briefly describes the two proposed terminal authentication protocol with complete illustrations. Section 3 describes the claim to show how secure the proposed protocol is. Section 4 shows experimental result with formal analysis by scyther tool. At last, we concludes our work and points out some future works in section 5.

2. Proposed Work

It has been observed in the literature survey that efficient solution of terminal authentication is not there although the work of authentication in IoT depends on certificate based authentication like works of Schmitt *et al.*[4], Hummen *et al.*[5] etc. However work of Schmitt *et al.* is applicable to range of things that gives additional advantage but there are very few non- certificate based solutions. The certificate based solutions tries to make existing security framework (e.g. - DTLS etc.) compatible to constrained device scenario[4,5]. They try to delegate some expensive operations to one dedicated server[5], that gives additional security loopholes (more prone to MITM from the dedicated server and should be trusted always). So we have tried to address these issues in the proposed work.

There are very few works we have found in literature survey, that specifically address the terminal authentication issues in IoT scenario. One of the existing work of terminal authentication[6] is based on equipment manufacturer assigned key and operator assigned key that limits the scalability issue. It also requires dedicated authentication server that increases its cost. So, our aim is to frame a terminal authentication protocol that address all issues of IoT like scalability, heterogeneity etc. in very less deployment cost but more secure.

Here, we have proposed the terminal authentication mechanisms (TAP 1 and TAP 2) for different kind of devices (*viz.* low end, high end, physical attack prone device etc.). Our proposed mechanisms are based on digital signatures and its variations like blind signature[8] (in limited sense, just for encryption/decryption with its public/private key of already encrypted packet) for one of the mechanism (TAP 1). Blind signature algorithms are basically used so far for user credentials retrieval[9], anonymous credential retrieval[10], single password authentication[11] etc. Blind signature algorithm is nothing but an extension of digital signature where receiver receives the signature on a message without revealing the message to the signer. So, we have used blind signature in one of the mechanism where we try to store our credentials to any storage server (Fig. 1). Our other proposed mechanism for terminal authentication (TAP 2) is based on simple digital signature.

Our terminal authentication protocol is based on the ETSI M2M architecture[1,7]. The working domain of this protocol is Device and Gateway Domain. The primary need in IoT is to authenticate genuine devices in smart home, smart healthcare, assets management, industrial automation etc. We cannot allow other devices to inappropriately access our domain by just joining it. So, our terminal authentication protocols (TAP) satisfies this requirement.

2.1 Proposed terminal authentication protocol 1 (TAP 1)

As there are several threats pointed out by contemporary work[4] and among them one of the major threats is extraction of security parameters. In any M2M scenario, generally devices are basically install in open environment and prone to physical attack like this. However, other mechanism are there to mitigate it like dedicated HSM (Hardware Security Module) installed on device but, it affects the scalability of the proposed mechanism and also increases cost (in Fig. 1, it is shown optionally as H/W). This protocol is also feasible to resource constraints devices given in[4], where storage is a major concern. So, here we are proposing generic terminal authentication protocol that tries to mitigate this kind of security problem by storing one of the authentication credentials to the storage gateway. The main feature of the proposed mechanism, it is not mandatory that storage gateway is trusted (that overcome the “trusted” assumption of previous works). However, we are assuming here that storage gateway is not able to do cryptanalysis and other sophisticated offline attack on stored credentials.

There are three entities participates in this protocol *viz.* M2M Device, M2M Gateway and Storage Gateway (the term "storage server" is also used interchangeably). Here functionality of Storage Server is to store the signature of the device without knowing it. Just to differentiate from M2M device, we call it as Storage Server. It could be implemented in any M2M Gateway or in any other capable device in local domain. This Protocol works in four phase:- Registration Phase, Storage Phase, Retrieval and Authentication Phase. This protocol is for centralized architecture where Gateway is the main authentication authority. Each device has to authenticate by Gateway before accessing the network domain or performing data transfer. All three participating entity have their public-private key pair like (dvk, dsk), (bvk_g, bsk_g), (bvk_s, bsk_s) for M2M device, M2M Gateway and Storage Server respectively. It has been shown in recent works[12] that public key cryptosystems (e.g. ECC) is fully applicable in IoT. Here we have assumed that key generation is already taken place. However, we can use symmetric keys also, if there is a provision of generate, distribute and store it securely. But as of now, we have used asymmetric keys in our experiments. Pictorial view of the proposed TAP 1 can be seen in Fig. 1 (here the assumption of one M2M Device is for abstraction purpose only otherwise there is no bondage on number of devices). Trivial features like Nonces, Message Ids are assumed there in TAPs, just for making the figures (1 & 2) concise, we have omitted it.

Main Idea. The protocol specification is as follows:

Device registration phase: It executes between M2M device and M2M Gateway.

Storage phase: Executes between M2M Device & Storage Server. After executing Registration Phase and Storage phase, **device will store {Blind, dsk} only**.

Retrieval Phase: Executes between Device & Storage server.

Authentication Phase: Executes between Device & M2M Gateway.

At **Device Side:-**
– Generates one Rand_string.
– Compute Hash (Rand_str) =H
– Perform E (H) $_{dsk}$ =Sig.
– Send it to Gateway in form of E {Sig., Device_Id, dvk} $_{bvkg}$
At **Gateway Side:-**
– Decrypt the received packet D {E {Sig., Device_Id, dvk} $_{bvkg}$} $_{bskg}$
Read Sig., Device_Id, dvk.
– Perform Decryption D {Sig.} $_{dvk}$
Stores H, Device_Id, dvk and **generates Reg.Id** and **also stores it.**
– Now, Gateway sends E {Reg.Id} $_{dvk}$ to device.
Device reads Reg.Id by decrypting D {E {Reg.Id} $_{dvk}$} $_{dsk}$

Algorithm 1. Registration Phase of the Proposed TAP

At Device Side:-
– Now, Device sends E {Sig, Device_Id, dvk} $_{bvks}$ & E {Reg.Id} $_{dvk}$ to storage server.
At Storage Server Side:-
– Decrypts D{E{Sig., Device_Id, dvk} $_{bvks}$} $_{bsks}$
– Now Performs Blind Signature on E{Reg.Id} $_{dvk}$ as E{E{Reg.Id} $_{dvk}$} $_{bsks}$
Blind.
– Stores {Sig., Device_Id} only in its database.
– Now, E {Blind} $_{dvk}$ will be sent to the Device.
– Now, flush the buffer.

Algorithm 2. Storage Phase of the Proposed TAP

At Device Side:-
– Sends E {Blind, Device_Id} $_{bvks}$
At Storage Server Side:-
– Decrypts D {E {Blind, Device_Id}$_{bvks}$ } $_{bsks}$
– Performs Inverse Blind as D {Blind}$_{bvks}$ = Blind' = E {Reg.Id}$_{dvk}$
– Searches {Sig.} with their respective Device_Id in its database.
– Now, sends {Sig.} + {Blind'} to the device.

Algorithm 3. Retrieval Phase of the Proposed TAP

At Device Side:-
– Decrypts D {Blind'}$_{dsk}$
– Prepares message – {Sig., Reg.Id, Device_Id} $_{bvkg}$ and sends it.
At Gateway Side:-
– Decrypts message- D{E{Sig., Reg.Id, Device_Id} $_{bvkg}$} $_{bskg}$
– From {Reg.Id, Device_Id} retrieves the stored {H, dvk}
– Now performs D{Sig.}dvk = H' then compares H' == H
– Then signals Accept/Reject to the device.

Algorithm 4. Authentication Phase of the Proposed TAP

2.2 Proposed terminal authentication protocol 2 (TAP 2)

Now, this proposed work is meant for the devices those are capable to store the authentication credentials securely. In this case we are assuming that all credentials are safe at device side and the attacker is not able to extract it. So, there is no need of separate storage server, therefore we have removed the storage entity and their communications of Terminal Authentication Protocol 1 (TAP 1). The Registration and Authentication part will remain same as TAP 1. However security part is still remain intact. As we have shown this in formal security analysis part of coming section. Again this proposed protocol is fully capable of supporting scalability and heterogeneity (refer Fig. 2).

2.3 Description of terminal authentication protocol

- There are three/two entities participates (for TAP 1 and TAP 2 respectively) in these protocols *viz.* M2M Device, M2M Gateway and Storage Server. Here functionality of Storage Server is to store the (Sig., Device Id) of the device (for TAP 1), and it need not to be trusted.
- Devices have prior information about public key of gateway and storage server but device's public key is not known to gateway and storage server unless it is sent by device explicitly.
- These protocols are for centralized architecture where Gateway is the main authentication authority.
- Each device has to authenticate by Gateway before accessing the network domain or performing data transfer.
- These Protocols works in four phase:- Registration Phase, Storage Phase, Retrieval & Authentication Phase (for TAP 1) and Two phase:- Registration Phase and Authentication Phase (for TAP 2).

2.4 Workings of terminal authentication protocol

The execution order of TAP will be from Registration, Storage, Retrieval & Authentication phase respectively (For TAP 2, Registration & Authentication phase). Basically Registration & Storage phase will execute at the time of bootstrapping (For TAP 2, Registration phase only). Which devices has to be registered in registration phase in what device & gateway domains is decided by the administrator/network maintenance staff. Retrieval phase could execute at the time of session initialization (for TAP 1). Authentication phase could execute any time in the session, based on requirement. Above all when to execute what is decided by network administrator/staff depends upon security needs.

3. Informal Threat Analysis

– *Security against passive eavesdroppers*

In this protocol, it is not possible to get any data since all exchange are encrypted with either bvk_g or dvk or bvk_s or dsk. So it is not possible to get H, Reg.Id etc.

– *Security against active impersonator*

If anyone impersonates device then it is not harmful because the protocol requires registration phase before authentication and also for authentication, Device id and Reg. Id both are required so one has to get/guess {dsk} & {Device_Id} of genuine device, that is not possible. There is no case of device impersonation in registration phase

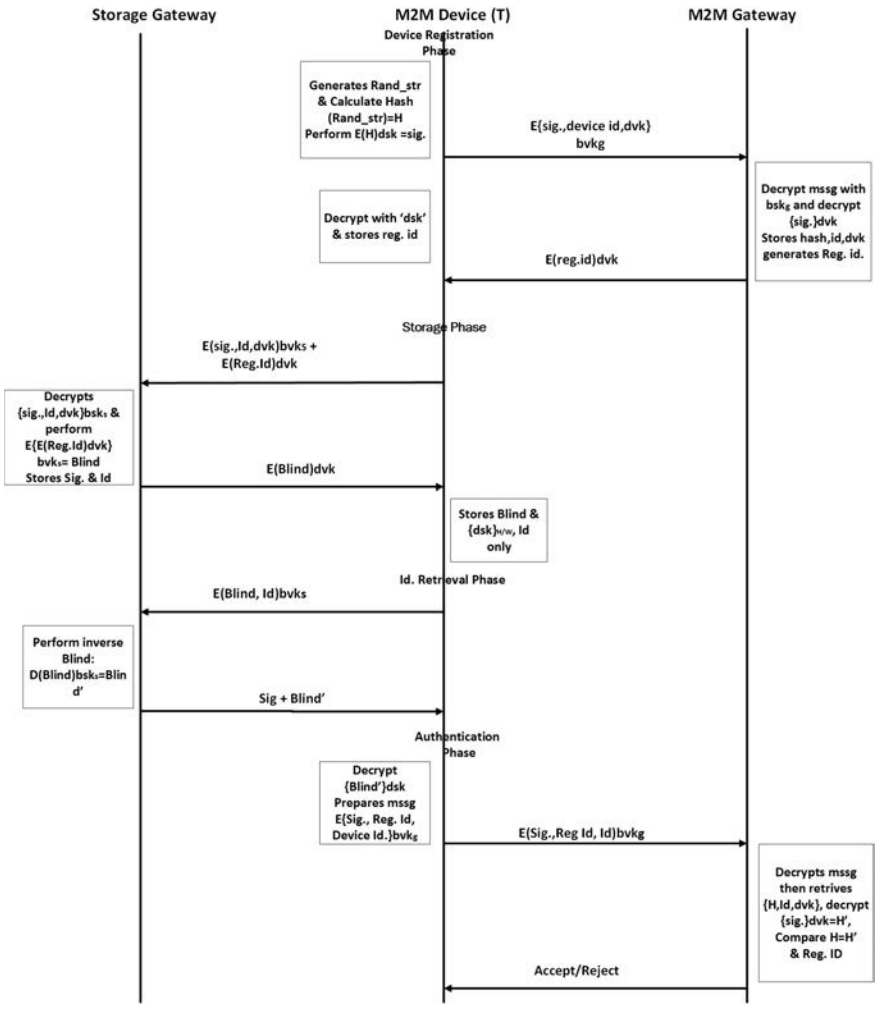


Fig. 1. Proposed Terminal Authentication Protocol 1 (TAP 1).

because at the time of initial bootstrap, which device has to join what domain is decided by human entity. There is no risk of Storage server impersonation because TAP is fully robust to untrusted storage server. It stores the Data in encrypted form which decryption key is not known to server and same is valid for storing Reg. Id.

– *Security against insider*

Devices cannot impersonate each other because {Sig.} of every device is already stored in gateway, so after mismatch rejection will happen. Device impersonation by other devices of same domain is not possible due to every device has to generate their own {Sig.} then it is stored in server. Here one more important assumption is that Device Ids is not known to each other or to the attacker. For device ids, we are assuming here as a pre-provisioned identifier but depending upon security and resource constraints, we could use other identifiers like M2M node identifier, M2M service identifier etc.

– *Physical Attacks may not possible*

Due to physical attacks like malicious extraction of security parameters, cloning of things etc. We have specifically proposed the terminal authentication protocol (TAP 1), where some authentication credentials are stored on storage server. If someone (attacker) tries to extract the credentials from the device then he/she cannot able to authenticate the

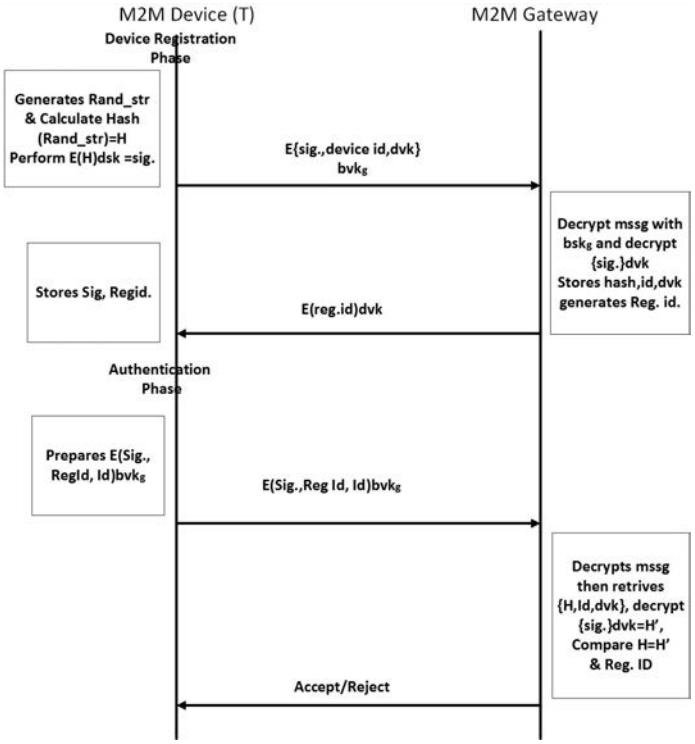


Fig. 2. Terminal Authentication Protocol 2 (TAP 2).

terminals because some of the authentication credentials (signature of the device) are stored on storage server. This assumption fails only if attacker is able to perform the attack simultaneously on storage server. That case does not arise in normal course and also if storage server lies on different domain. There is no adverse effect on proposed scheme if storage server lies on different domain if provided efficient addressing mechanism to communicate.

– *Chances of brute force attacks are negligible*

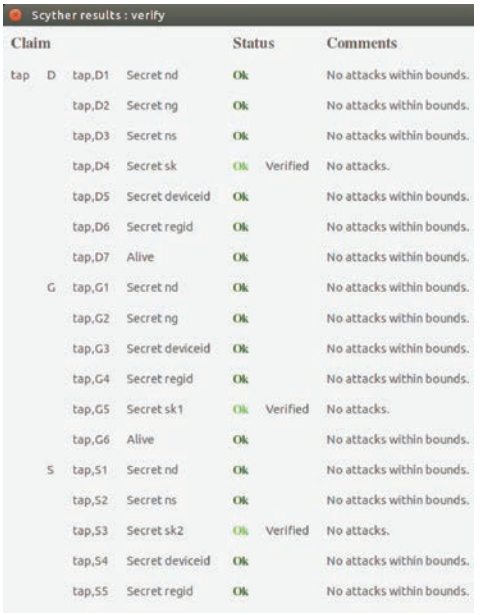
Here, the basis of device authentication is hash and hash is calculated by purely random generated string of each device. So brute force attack on gateway side and storage server side is not possible. However, we can further restrict the unsuccessful authentication attempt limited to some fixed number of times (e.g. 3).

4. Experiments and Results

We have performed two types of experiments:- First we have done a formal analysis by model based formal analyser called "Scyther"[14] and then we have implemented the TAP in real time in our dept. lab and concludes its efficiency and deployability in subsection 4.3.

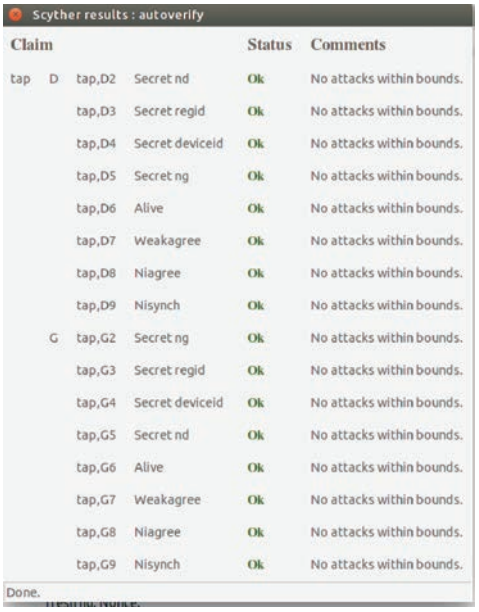
4.1 Formal security analysis

We have written the spdl script for both proposed protocols and compiled it on scyther. The output (Fig. 3 & 4) has shown that our proposed protocols are free from attacks. We have shown the two output window of protocol verification and protocol claims and both output are showing the results those are free from attacks within the bound. We have analysed the protocols with default round of 5 in scyther.


Scyther results : verify

Claim				Status		Comments
tap	D	tap,D1	Secret nd	Ok		No attacks within bounds.
		tap,D2	Secret ng	Ok		No attacks within bounds.
		tap,D3	Secret ns	Ok		No attacks within bounds.
		tap,D4	Secret sk	Ok	Verified	No attacks.
		tap,D5	Secret deviceid	Ok		No attacks within bounds.
		tap,D6	Secret regid	Ok		No attacks within bounds.
		tap,D7	Alive	Ok		No attacks within bounds.
	G	tap,G1	Secret nd	Ok		No attacks within bounds.
		tap,G2	Secret ng	Ok		No attacks within bounds.
		tap,G3	Secret deviceid	Ok		No attacks within bounds.
		tap,G4	Secret regid	Ok		No attacks within bounds.
		tap,G5	Secret sk1	Ok	Verified	No attacks.
		tap,G6	Alive	Ok		No attacks within bounds.
	S	tap,S1	Secret nd	Ok		No attacks within bounds.
		tap,S2	Secret ns	Ok		No attacks within bounds.
		tap,S3	Secret sk2	Ok	Verified	No attacks.
		tap,S4	Secret deviceid	Ok		No attacks within bounds.
		tap,S5	Secret regid	Ok		No attacks within bounds.

Fig. 3. Scyther's Verification output for TAP 1.


Scyther results : autoverify

Claim				Status	Comments
tap	D	tap,D2	Secret nd	Ok	No attacks within bounds.
		tap,D3	Secret regid	Ok	No attacks within bounds.
		tap,D4	Secret deviceid	Ok	No attacks within bounds.
		tap,D5	Secret ng	Ok	No attacks within bounds.
		tap,D6	Alive	Ok	No attacks within bounds.
		tap,D7	Weakagree	Ok	No attacks within bounds.
		tap,D8	Niagree	Ok	No attacks within bounds.
		tap,D9	Nisynch	Ok	No attacks within bounds.
	G	tap,G2	Secret ng	Ok	No attacks within bounds.
		tap,G3	Secret regid	Ok	No attacks within bounds.
		tap,G4	Secret deviceid	Ok	No attacks within bounds.
		tap,G5	Secret nd	Ok	No attacks within bounds.
		tap,G6	Alive	Ok	No attacks within bounds.
		tap,G7	Weakagree	Ok	No attacks within bounds.
		tap,G8	Niagree	Ok	No attacks within bounds.
		tap,G9	Nisynch	Ok	No attacks within bounds.

Done.

Fig. 4. Scyther's Verification output for TAP 2.

4.2 Protocol implementation

We have used Python 2.7 with Pycrypto 2.4 library on Ubuntu 14.04 Linux for implementing the proposed protocol. Hardware configuration was Intel i5 3rd gen processor (1.4 GHz) with 2GB RAM. We have written the separate script for M2M Gateway, M2M Device and M2M Storage Server for TAP 1 (M2M Device and M2M Gateway for TAP 2). For generating RSA keys, we have written a separate python script. The cryptographic specification of RSA Keys is 2048 bit, SHA 1 (for hash) is 256 bit and PKCS v1.5 are used through Pycrypto in our implementation. These specification are not mandatory, just for experimental purposes we have used 'heavy' size cryptographic functions. As our proposed work is meant for diverse range of devices from traditional computing to tiny sensors. So, we can easily switch to lighter version of these functions or we can choose the alternate functions according to the configuration of the devices. Our main aim is to show its efficiency as per Programme Execution Time, CPU Time and Wall Clock Time through timeit.timeit(), time.clock(), time.time() modules of python.

4.3 Experimental setup and results

We have executed the script in three different computers connected through LAN in our university lab by establishing connection over socket. The transport protocol we have used is TCP, again this is for experimental purpose. It could also easily implementable on UDP and other lighter version protocols. After successfully implemented experimental setup for TAP 1 and TAP 2, we can easily say that proposed work is very much efficient in traditional computing systems with average code execution time are 0.0006s for device, 0.0021s for server and 0.0019s for storage server in TAP 1 and average code execution time are 0.0014s for device and 0.0019s for server in TAP 2. Average CPU time are 0.0627s for device, 0.0300s for gateway, 0.0408s for storage server in TAP 1 and 0.0340s for device, 0.0400s for gateway in TAP2 (complete lists are given in Appendix). We have also found during the experiment that maximum 22 bytes of data are transferred (if we use such heavy keys then). So, it also generates very low data traffic and consumes less resources for data processing. The main objective of our experiment was to show its deployability and efficiency. Through experiments, we have successfully achieved both objective through implementing and calculating different execution time.

5. Conclusions and Future Work

In this work, we have proposed the new terminal authentication protocols for M2M systems basically for internet of things. As in IoT scenario, Scalability and Heterogeneity are the main concern, so our proposed work satisfies this requirement. Like previous work, this mechanism does not depends on some pre-installed keys or some other tokens. All credentials are randomly generated, despite of nature and configurations of the devices.

However our work depends on public/private key pair and device should also be capable of performing digital signatures. These are the trivial features of nowadays devices and few proposals has been made already in IoT scenario like[12,13] etc. So, our proposed work supports heterogeneity. The proposed work also supports scalability because there are at least two phase (for TAP 1, four phases) *viz.* registration phase and authentication phase in the mechanism that enable it to perform the authentication of as many devices as possible.

As we have implemented the proposed work with large size of cryptographic keys, that even shows the very less code execution time and CPU time. This concludes that the protocol is very much efficient and fast.

We will try to propose mutual authentication mechanism for terminals. We show its variant of symmetric keys by experiments. We try to integrate these protocols to some specific resource constraint application protocol like CoAP etc.

Acknowledgements

We are grateful to *Dr. Maniklal Das*, *DA-IICT Gandhinagar India* for his fruitful suggestions in this work.

Appendix

Following tables shows the different timings of TAP 1.

Table 1. Different timings of the device.

Execution Serial No.	Code Execution Time (s)	CPU Time (s)	Wall Clock Time (s)
1	0.0006	0.0622	0.1076
2	0.0011	0.0626	0.1093
3	0.0011	0.0619	0.1094
4	0.0001	0.0656	0.1101
5	0.0006	0.0614	0.1069
6	0.0005	0.0646	0.1125
7	0.0002	0.0610	0.1086
Avg.	0.0006	0.0627	0.1092

Table 2. Different timings of the gateway.

Execution Serial No.	Code Execution Time (s)	CPU Time (s)	Wall Clock Time (s)
1	0.0003	0.04	4.2537
2	0.0017	0.04	1.8645
3	0.0051	0.03	2.1341
4	0.0051	0.03	1.9398
5	0.0015	0.04	1.5577
6	0.0003	0.04	1.6624
7	0.0007	0.04	1.6462
Avg.	0.0021	0.03	2.1512

Table 3. Different timings of the storage server.

Execution Serial No.	Code Execution Time (s)	CPU Time (s)	Wall Clock Time (s)
1	0.0004	0.0391	4.0264
2	0.0004	0.0421	1.5108
3	0.0002	0.0363	1.9777
4	0.0020	0.0424	1.6860
5	0.0009	0.0414	1.4566
6	0.0005	0.0432	1.5672
7	0.0089	0.0414	1.5089
Avg.	0.0019	0.0408	1.9619

Following tables shows the different timings of TAP 2.

Table 4. Different timings of device.

Execution Serial No.	Code Execution Time (s)	CPU Time (s)	Wall Clock Time (s)
1	0.0019	0.0356	0.1056
2	0.0001	0.0321	0.0716
3	0.0004	0.0324	0.0697
4	0.0040	0.0360	0.0757
5	0.0016	0.0340	0.0733
6	0.0006	0.0311	0.0692
7	0.0018	0.0340	0.0717

Table 5. Different timings of the storage server.

Execution Serial No.	Code Execution Time (s)	CPU Time (s)	Wall Clock Time (s)
1	0.0032	0.0300	0.1720
2	0.0032	0.0300	0.2882
3	0.0007	0.0200	0.2622
4	0.0014	0.0300	0.5912
5	0.0029	0.0300	0.4082
6	0.0008	0.0400	1.1092
7	0.0011	0.0400	0.4471

References

[1] J. Holler, V. Vlasios Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand and D. Boyle, From Machine to Machine to the Internet of Things, *Academic Press, Elsevier*, (2014).

[2] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu and D. Qiu, Security of the Internet of Things: Perspectives and Challenges, *Journal of Wireless Network*, Springer Science + Business Media New York, (2014).

[3] M. Chen, J. Wan, S. Gonzalez, X. Liao and V. C. M. Leung, A Survey of Recent Development in Home M2M Networks, *First Quarter: IEEE Communications Surveys & Tutorials*, (2014).

[4] C. Schmitt and B. Stellar, Two Way Authentication in IoT, *IETF Internet Draft*, (2014).

[5] R. Hummen, J. H. Ziegeldorf, H. Shafaghy, S. Razay and K. Wehrle, Towards Viable Certificate-based Authentication for the Internet of Things, In: *ACM HotWiSec*, (2013).

[6] F. Gao, W. The Silver Star, T. Tian and Z. Yunwen, Machine to Machine Terminal Authentication Systems and M2M Terminal Authentication Method, Chinese Patent CN 201010151160, (19 October 2011).

[7] ETSI TS 102 690 v2.1.1, http://www.etsi.org/ (2013).

[8] D. Chaum, Blind Signatures for Untraceable Payments, In: *CRYPTO*, (1982).

[9] X. Boyen, Hidden Credentials Retrieval from a Reusable Password, In: *ACM ASIACCS*, (2009).

[10] F. Baldimtsi and A. Lysyanskaya, Anonymous Credentials Light, In: *ACM ASIACCS*, (2013).

[11] T. Acar, M. Belenkiy and A. Kupcu, Single Pasword Authentication, *Journal of Computer Networks, Elsevier*, vol. 57, pp. 2597–2614, (2013).

[12] S. Guicheng and Y. Zhen, Application of Elliptic Curve Cryptography in Node Authentication of Internet of Things, In: *Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE*, (2013).

[13] F. Razzak, Spamming the Internet of Things: A Possibility and its probable Solution, In: *MobiWIS. Procedia Computer Science, Elsevier*, vol. 10, pp. 658–665, (2012).

[14] Scyther User Manual, (2013), [online]. Available: http://people.inf.ethz.ch/cremersc/scyther/index.html